N72-24211

NASA CONTRACTOR
REPORT

NASA CR-61379

THE MSFC UNIVAC 1108/EXEC 8 SIMULATION MODEL

By T. Williams, F. M. Richards,
J. E. Weatherbee, and L. K. Paul
Computer Sciences Corporation
Field Services Division, Aerospace Systems Center
8300 S. Whitesburg Drive
Huntsville, Alabama 35802

March 31, 1972

Prepared for

NASA-GEORGE C. MARSHALL SPACE FLIGHT CENTER
Marshall Space Flight Center, Alabama 35812

TECHNICAL REPORT STANDARD TITLE PAGE

1. REPORT NO. NASA CR-61379	2. GOVERNMENT ACCESSION NO.	3. RECIPIENT'S CATALOG NO.
4. TITLE AND SUBTITLE The MSFC UNIVAC 1108/EXEC 8 Simulation Model		5. REPORT DATE March 31, 1972 6. PERFORMING ORGANIZATION CODE
7. AUTHOR (S) T. Williams, F.M. Richards, J.E. Weatherbee, L.K. Paul		8. PERFORMING ORGANIZATION REPORT #
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Sciences Corporation Field Services Division, Aerospace Systems Center 8300 S. Whitesburg Drive, Huntsville, Alabama 35802		10. WORK UNIT NO. 11. CONTRACT OR GRANT NO. NAS8-21805
12. SPONSORING AGENCY NAME AND ADDRESS National Aeronautics and Space Administration Washington, D.C. 20546		13. TYPE OF REPORT & PERIOD COVERED NASA Contractor Report 14. SPONSORING AGENCY CODE
15. SUPPLEMENTARY NOTES		

16. ABSTRACT

This report defines a model which simulates the MSFC Univac 1108 Multiprocessor System. The hardware/operating system is described to a level of detail necessary to enable a good statistical measurement of the system behavior to be made.

The performance of the 1108 is evaluated by performing twenty-four different experiments designed to locate system bottlenecks and also to test the sensitivity of system throughput with respect to perturbation of the various Exec 8 scheduling algorithms.

The model is implemented in the General Purpose System Simulation (GPSS) language and the techniques described can be used to assist in the design, development and evaluation of new multiprocessor systems.

17. KEY WORDS Computer System Simulation Simulation Model Simulator UNIVAC 1108 EXEC 8	18. DISTRIBUTION STATEMENT Bobby Hodges B. Hodges Computer Systems Division MSFC/Computation Laboratory Unclassified-unlimited

19. SECURITY CLASSIF. (of this report)	20. SECURITY CLASSIF. (of this page)	21. NO. OF PAGES	22. PRICE
Unclassified	Unclassified	59	$3.00

MSFC - Form 3292 (May 1969)

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

UNUSUAL TERMS

1.	GPSS HELP Block	-	GPSS command which transfers control to the user provided HELP subroutine.
2.	GPSS Parameter	-	Attributes of GPSS transaction.
3.	GPSS Transaction	-	The unit of traffic processed by the simulator.
4.	MAXOPEN	-	UNIVAC 1108/EXEC 8 system parameter which limits the number of concurrent active runs.
5.	THRPUT Program	-	Software monitor of U-1108/8

NONSTANDARD ABBREVIATIONS

1. U-1108 - UNIVAC 1108
2. U-1108/8 - UNIVAC 1108/EXEC 8
3. R.E.S.T. - Run Entered System Time

ACKNOWLEDGMENT

The authors wish to acknowledge Dr. H. Kerner (S&E-COMP-C), the initiator of the project, for his help and guidance during the many discussions and planning sessions held during the course of the work. Thanks is also due to the Project Monitor, Mr. B. Hodges (S&E-COMP-C), for his helpful contributions made throughout the project and in preparation of this report.

SUMMARY

To gain experience in simulating a large multiprocessor complex, a deterministic throughput model of the MSFC UNIVAC 1108 has been developed. The model simulates the MSFC UNIVAC 1108 when processing a stream of programs under the EXEC 8 operating system.

As well as determining the system throughput for a given load, the simulator allows the interaction between the various user programs and the executive operating algorithms to be examined in detail. Hence, bottlenecks and hardware/software mismatches can be identified and experimentally resolved.

The simulator input is a stream of tasks generated from the system accounting tape and the console log. These tasks represented by attributes such as priority, core and central processing unit (CPU) time requirements, are used to construct an input runstream having the same characteristics as the actual runstream. This runstream is stored on a tape (JOBTAPE) and is used as the driving mechanism for the simulator.

The simulation model is coded in the general purpose simulation language, GPSS, and FORTRAN. GPSS was chosen for its suitability to throughput-type investigations.

The milestones of each run's life cycle, as it moves through the UNIVAC 1108/EXEC 8 (U-1108/8) system, involve input, facility acquisition, core allocation, processor access, and finally program generated output. These milestones are achieved via the various EXEC 8 components: Card Read Symbiont, Coarse Scheduler, Dynamic Allocator, Dispatcher, and the Print and/or Punch Symbionts.

Each of these modules has been implemented in the simulator and interconnected to form a deterministic model of the overall U-1108/8 system.

An important component of the model is the load imposed upon the system by the execution of EXECUTIVE tasks. This represents the collection of all tasks which EXEC 8 must perform during the course of processing a given stream of user tasks. Since neither the Accounting Log Tape nor the THRPUT program collects sufficient data to provide precise inputs for the simulator in this area, it was arbitrarily decided to create one EXEC task for each of the user tasks within a given runstream. The primary function of such EXEC tasks is to place an additional load upon the CPU's and I/O subsystems within the model, thereby delaying the execution of user tasks as in the actual system.

For each run processed by the simulator, two times are recorded: (a) The run-entered-system-time, T_e, which is the time the run card is read by a card-reader, and (b) the termination time, T_t,

the time at which the run completes execution. Both these times are recorded on the system log. The run-entered-system-time may be used as the parameter which provides the arrival pattern of runs to the simulator. T_t is then determined for each run processed by the simulator and can be compared with the times logged on the system accounting tape.

The simulator may also be monitored at many internal points to give such system parameters as core utilization, CPU utilization, channel utilization, channel waiting times, etc.

The model was verified by comparing its measured performance with that of the actual U-1108/8 system as recorded on the accounting tape.

Using a generalized job-mix, a series of experiments was carried out on the simulator. These experiments were designed to locate system bottlenecks and also to test the sensitivity of system throughput with respect to perturbation of the various Exec 8 scheduling algorithms. A major system bottleneck was located, and two different solutions for its removal are identified.

SECTION I. INTRODUCTION

To gain experience in simulating a large multiprogramming/multiprocessor computer environment, a deterministic throughput model of the MSFC UNIVAC 1108 system has been developed. This model simulates the MSFC UNIVAC 1108 when processing a stream of runs under the EXEC 8 operating system.

The simulator allows the interaction between the various user programs, the executive scheduling algorithms, and the configuration hardware elements to be examined in detail. Because of this capability, algorithm and/or hardware changes may be studied and evaluated with the simulator and not require actual system implementation.

The simulator input is a stream of runs sampled from actual installation accounting data. This input-stream is defined in terms of attributes such as run priority, number of tapes, task core requirements, task CPU time requirements, and the number of task I/O references (see Figure 3).

The simulation model, presently implemented on the UNIVAC 1108, is coded in the general purpose simulation language, GPSS, and FORTRAN.

Figure 1 describes the MSFC UNIVAC 1108 hardware configuration.

A. Overall Flow of Exec 8

Figure 2 depicts the abbreviated logic flow for a run as it moves through the UNIVAC 1108/EXEC 8 (U-1108/8) system. The milestones of each run's life cycle involve run card-deck input, facility acquisition, core allocation, and processor utilization. These milestones are achieved via the various EXEC 8 components: Input Symbionts, Coarse Scheduler, Dynamic Allocator and Dispatcher, respectively. EXEC 8 also supervises all I/O requests to and from the peripheral subsystems. Each run arrives in the form of a punched card deck, the card images being transferred to a run file on mass storage as the deck is input on either the Communications Terminal Module Controller (CTMC) or Central Site Card Reader. The runs are ordered in a run queue according to run priority.

The run having the earliest Run Entered System Time (R.E.S.T.) within the highest priority class of the run queue is examined for the purpose of acquiring the necessary mass storage and tape drives which will be required during execution. This examination is provided by the Coarse Scheduler, assignment being made only when all such facility requirements can be met simultaneously. If this condition cannot be met the next run within that priority class is considered by the

Coarse Scheduler. No run from a lower priority class can request facilities until the higher priority class has been satisfied. When the required facilities are assigned the run becomes eligible to enter the active state. This active state is entered only if one of the following criteria can be satisfied:

1) a. The active state limit (MAXOPEN) has not been exceeded, and

   b. no tasks are in the core queue.

2) The run is of demand status.

Once active, the facilities are allocated to the run and task 1 enters the core queue, where the Dynamic Allocator attempts to assign the I-bank and D-bank core requirements. Since tasks of multiple runs exist in the core queue concurrently, tasks are considered by the Dynamic Allocator in order of their priority. Upon allocation of core the task proceeds to the CPU queue. However, if core requirements cannot be met, the task remains in the core queue pending the release of core by the termination of an active task. The functions of the Dynamic Allocator are illustrated in Figure 7.

The Dispatcher is the routine through which EXEC 8 accomplishes its time-sharing processes. The Dispatcher uses an eight-level switch list (see Figure 8) to coordinate the activities of all the tasks which are resident in core. This mechanism allows CPU's to be assigned to and released from specific task execution as various events and contingencies arise (e.g., I/O initiations).

A task will lose control of a CPU in favor of another task at the currently lowest switch list level either upon expiration of a set time quantum, completion of the task's CPU requirements, or by voluntarily releasing control of a CPU while awaiting completion of an I/O request. On completion of the I/O request the task will enter the CPU queue at the lowest switch list level for which it is eligible.

The Dispatcher switches control according to the current structure of the list, always yielding to a task at the currently lowest switch list level.

The CPU time quantum has a value of 8 milliseconds for level 1, and is doubled for each subsequent level up to a maximum of 1.024 seconds at level 8.

Batch-type runs enter the list at level 2.

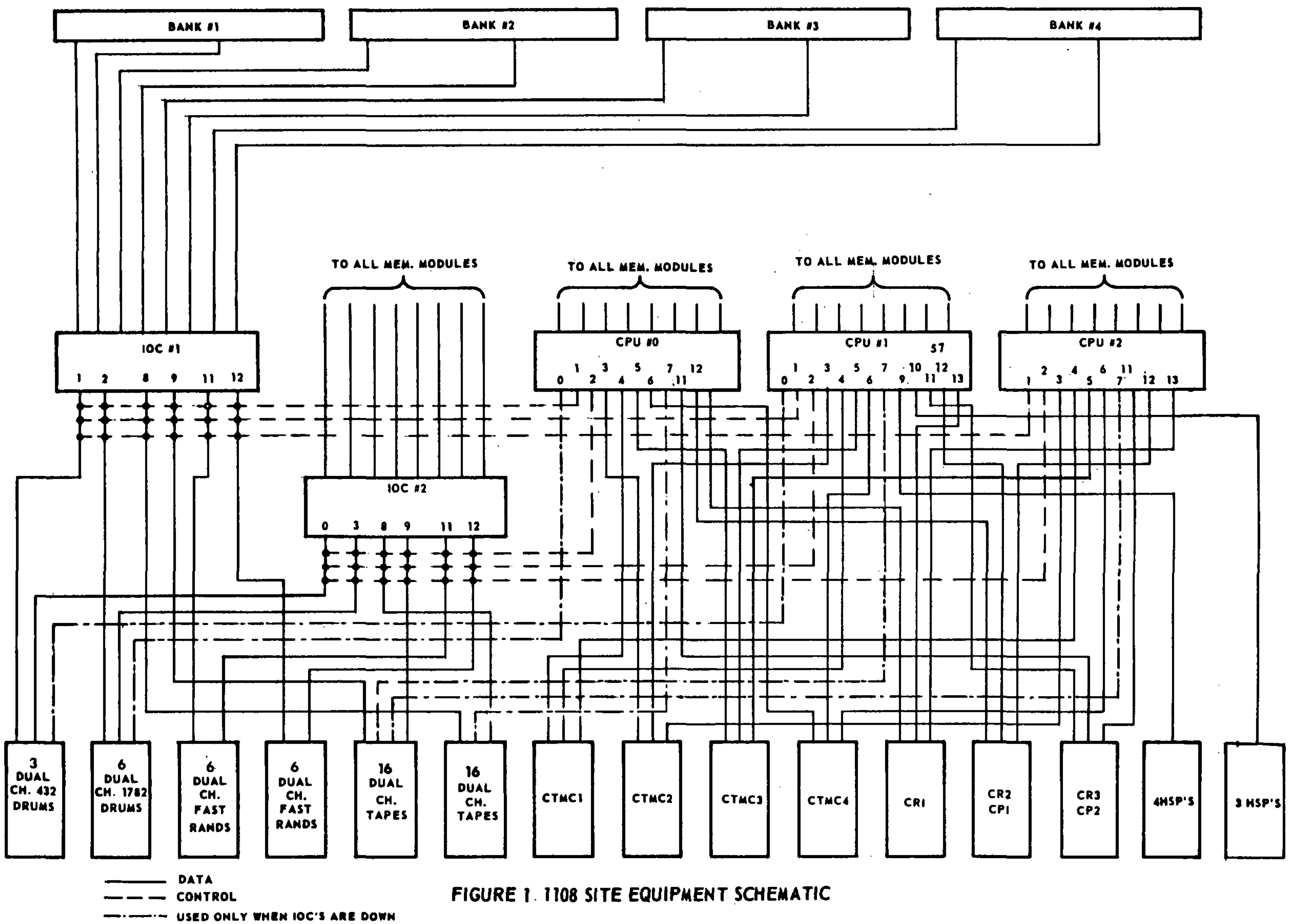


FIGURE 1. 1108 SITE EQUIPMENT SCHEMATIC

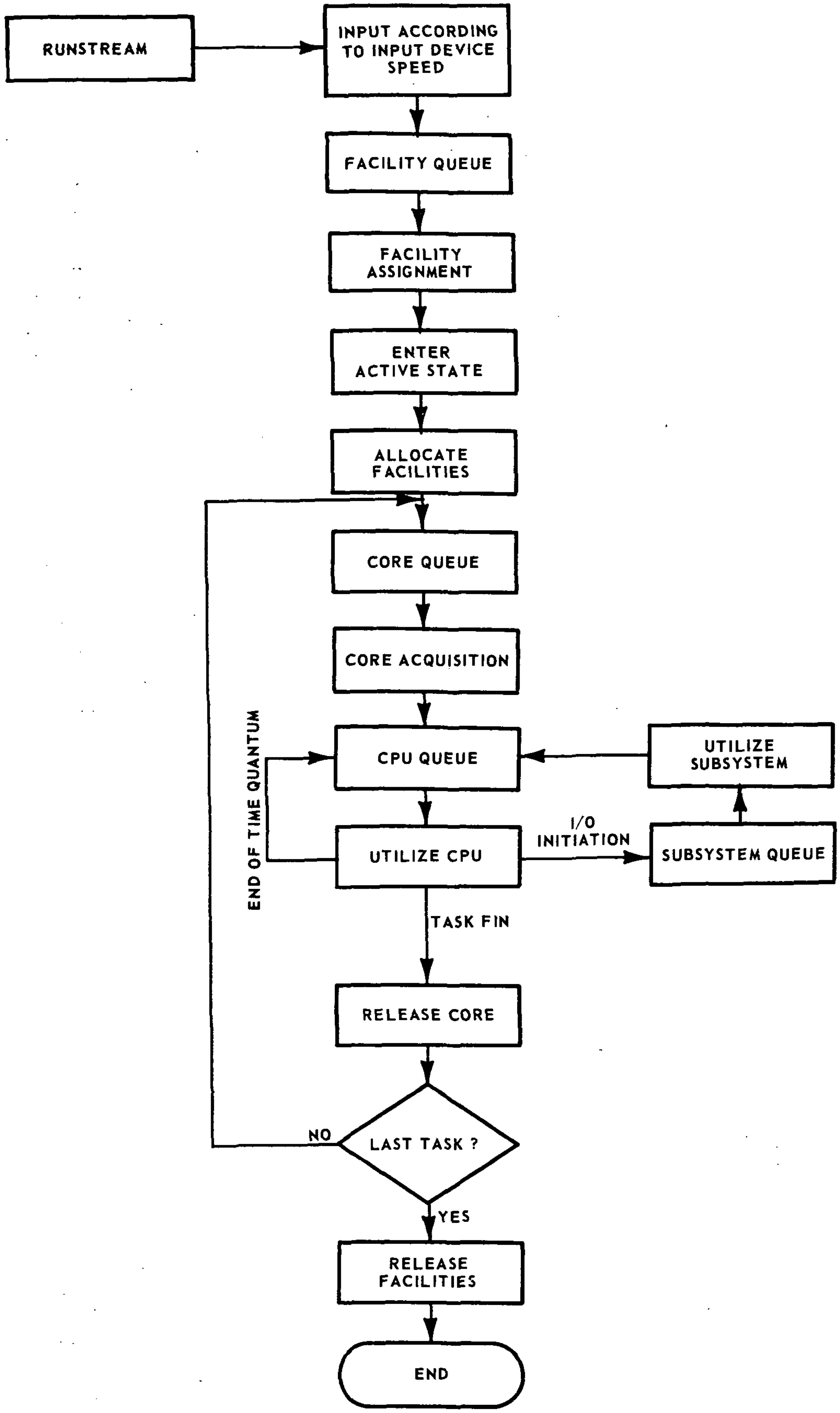


FIGURE 2. UNIVAC 1108/EXEC 8 – LOGIC FLOW

SECTION II. SIMULATION OF THE U-1108/8 SYSTEM

The preceding section summarized the U-1108/8 system and the remainder of the report describes how this system was modelled in order to obtain a statistical measure of system throughput.

In order to illustrate the level of detail of the simulation it is appropriate at this point to describe the method of synthesizing a mix of jobs which form the workload to be processed by the simulated system.

A. The User Programs

Actual data collected from the MSFC U-1108 accounting tapes and the corresponding console typewriter facility assignment records are used to fabricate realistic runstream inputs to the simulation model. This section describes the construction of this runstream. A computerized procedure has been developed to digest accounting tapes and create run tables over preselected time intervals corresponding to the three shifts operated at MSFC. The runstream is then constructed by randomly sampling the run table corresponding to a particular shift. The set of run attributes includes:

1) Priority
2) Number of input cards
3) Number of tasks
4) Input mode (on-site or remote)
5) Number of tapes
6) Demand status

The set of task attributes includes:

1) I-bank core requirements
2) D-bank core requirements
3) CPU time requirements
4) Number of I/O references

These attributes differ markedly from shift to shift due to the different nature of the data processing performed and hence three run tables are necessary.

The runstream information is contained in an input file to the model and the tasks of each run are represented by records of the file. At run "creation time" the run and task attributes are stored in the parameters of the GPSS transaction which will represent the task in the model.

Figure 3 describes the task parameter set.

PARAMETER	DESCRIPTION
1	Run Identification Number
2	Task Identification Number
3	Number of Tapes Required
4	Total Number of Tasks in Run
5	Run Priority
6	Demand Status:
	1 $\Rightarrow$ Demand, 0 $\Rightarrow$ Not Demand
7	Number of Input Cards to Run
8	Run's Input Mode:
	0 $\Rightarrow$ Central Site, 1 $\Rightarrow$ Remote Terminal
9	Task I-Bank Core Requirements
10	Task D-Bank Core Requirements
11	Number of I/O References
13,14	Task CPU Time (in milliseconds)

FIGURE 3. TASK PARAMETER SET

B. Executive Overhead

An important component of the simulator is the load imposed upon the system by the execution of Executive system tasks. For convenience, this will be referred to as EXEC overhead" and represents the collection of all tasks which EXEC 8 must perform during the course of processing a given stream of user tasks. Since insufficient system data is recorded to provide precise inputs to the simulation in this area it was arbitrarily decided to create one EXEC task for each of the user tasks within a given runstream. The CPU time required by the EXEC tasks is a variable which may be adjusted to represent various EXEC-to-user ratios. For the simulation work described in this report this ratio was fixed at 0.5. The primary effect of such EXEC tasks is to place an additional load upon the CPU's and I/O channels, thereby delaying the execution of user tasks. The core required by the Executive system consists of two reserved blocks, one at the low address area of the I-bank region and the other at the high address end of the D-bank section (see Section III-B).

C. Input/Output

The MSFC U-1108 configuration has six dual-channel I/O control units as shown on the site schematic of Figure 1. These control data transmission to and from the mass storage devices (drums and tapes).

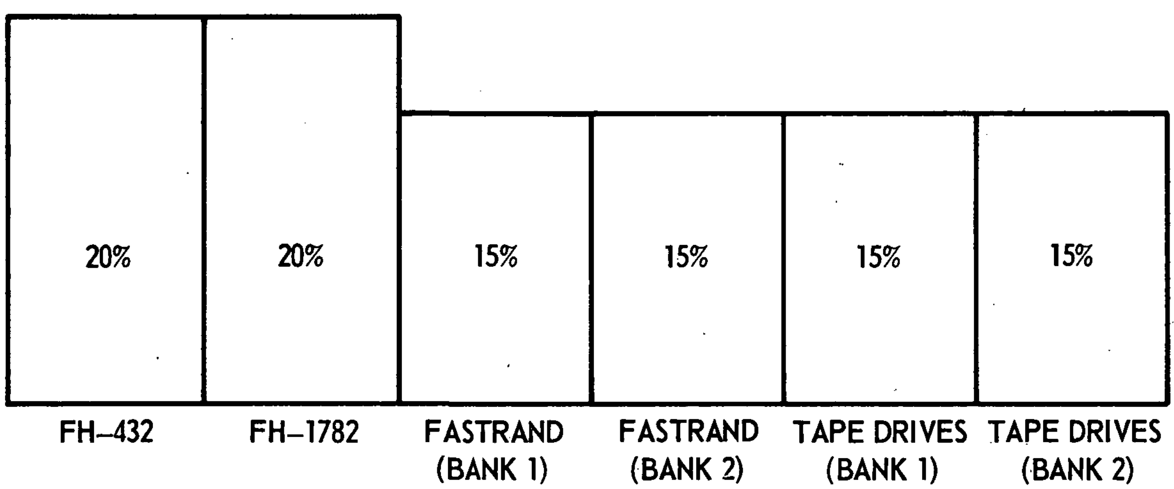
Utilization statistics for each of these control units were obtained from UNIVAC's THRPUT program and are input to the model. When an I/O request is dispatched to one of these dual channel control units, it enters a FIFO queue for that control unit; and when the request is serviced, it is assigned to one of the two channels.

Separate control unit utilization functions are allowable within the model for the user and the Executive I/O's, respectively. However, the common utilization distribution shown in Figure 4 is currently used for both user and Executive I/O's.

I/O subsystem service times (access time plus transmission time) are also input to the model for each control unit. These service time statistics, available from the THRPUT programs, are in the form of an average value for each control unit.

The Dispatcher section (see Section III-C) is used to control the mechanism which results in the I/O initiation.

Frequency of initiation for the user I/O's is determined as follows:



PERCENTAGE OF I/O'S TO U–1108 SUBSYSTEMS

FIGURE 4. I/O DISTRIBUTION FUNCTION

Each task has associated with it a number of I/O references obtained from accounting tape statistics. This number of I/O initiations will occur during the life of the task and are assumed to be periodic with respect to the task's CPU time. Hence, if a task has n I/O's and requires m milliseconds of CPU time, an I/O period $p = m/n$ milliseconds is computed. For the cases where $p < 1$, p is set to 1, since the model's basic time unit is 1 millisecond.

Frequency of initiation for Executive I/O's, however, is handled in a different manner; at the model user's option an arrival function may be specified and all Executive tasks will conform to this initiation pattern. Current implementation utilizes an average EXEC I/O initiation rate whose value was determined by experiment with the simulator. The EXEC I/O initiation rate selected was that which made the simulated CPU utilization figure match that of the actual system as given by the THRPUT program.

The "number of I/O references" mentioned above is a measure of physical I/O activity on the mass storage devices. Since, however, the periodic frequency of the I/O's is determined by the executing programs, a blocking factor on the physical I/O's is allowed to convert them to program-oriented or logical I/O's.

SECTION III. DETAILED SIMULATOR MODULES

A. Coarse Scheduler

The Coarse Scheduler is an executive function responsible for the assignment of peripheral devices and the introduction of new runs into the operating environment. The only peripheral devices which the simulator is concerned with, for purposes of allocation, are the tape drives. Figure 6 presents the logic flow for the Coarse Scheduler. According to run type there are two paths through the Coarse Scheduler. Those runs of demand status which enter the run queue are allowed to enter the active state immediately if their facilities are available. Runs of batch status must satisfy the two conditions:

1) The Maxopen limit would not be exceeded, and

2) the core queue must be empty.

That batch run with the earliest Run Entered System Time (R.E.S.T.) in the highest priority class is selected. If its facility requirements can be met it enters the active state and its facilities are allocated. If its facility requirements cannot be met, it returns to the run queue and the run with the next earliest R.E.S.T. within the highest priority class attempts to procure its facilities. This procedure goes on until a run is able to satisfy its facility requirements.

B. Dynamic Allocator

Memory is acquired on a per task basis according to R.E.S.T. within a priority class. The Dynamic Allocator has the function of efficiently assigning core space and maintaining a memory map. This includes the tasks of allocating available memory when requested and making it available for reallocation when released.

Main storage on the U-1108 is configured as shown in Figure 5. There are four banks of memory, each consisting of 65,536 words. These banks, although subdivided into odd/even storage locations to facilitate interleaving operations, constitute contiguous storage of instructions (I-bank) from location 0 upward, and storage of data (D-bank) from location 262,143 downward. Core is allocated from these extremities toward the middle by a "collision" process. EXEC 8 operating system routines reside in locations 0 up to 24,590 (I-bank) and from 217,111 up to 262,143 (D-bank). The Dynamic Allocator insures orderly utilization of the "gap" from 24,591 through 217,110.

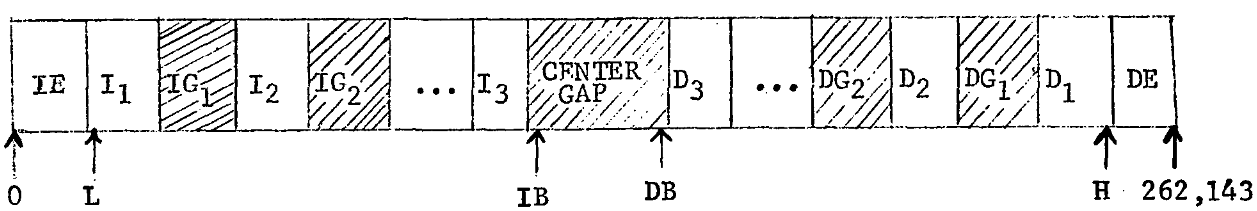
BANK #1		BANK #2		BANK #3		BANK #4	
065,534	065,535	131,070	131,071	196,606	196,607	262,142	262,143
⋮	⋮	⋮	⋮	⋮		⋮	⋮
2	3	065,538	065,539	131,074	131,075	196,610	196,611
0	1	065,536	065,537	131,072	131,073	196,608	196,609
I-Bank						D-Bank	

FIGURE 5. U-1108 STORAGE CONFIGURATION

Figure 7 shows the abbreviated logic flow for the Dynamic Allocator. Basically, tests are made to determine if both the I- and D-bank requirements can be met simultaneously. If so, the requested core is allocated, and the core map updated.

Following completion of a task on the CPU, its core is released via the Dynamic Allocator, and the core map is updated.

To facilitate precise discussion of the dynamic allocation of memory, consider the following representation for the 262,144 words of memory.



where,

IE: I-bank requirements of resident executive.

I_j: I-bank core requirements for some task.

L: Lowest assignable address for user tasks.

IG_j: Unassigned core. Right contiguous to I_j. (≥ 0)

IB: Current upper bound of I-bank.

DB: Current lower bound of D-bank.

D_j: D-bank core requirements for some task.

DG_j: Unassigned core. Left contiguous to D_j. (≥ 0)

H: Highest assignable address for user tasks.

DE: EXEC 8 variable data space and permanent data storage.

a) IB is either coincident with or has a value less than that of DB.

b) A center gap exists if IB<DB. This area would be available for assignment as either I-bank or D-bank.

c) From their definitions, IB and DB are obviously floating boundaries.

The allocation/release algorithm is coded in FORTRAN and is accessible to GPSS through the GPSS HELP block.

A search is made for the smallest IG such that $I_j \leq IG$. If such an IG cannot be found, the allocator rejects the request, sets the appropriate flag, and returns.

The smallest DG such that $D_j \leq DG$ is then searched for. If such a DG cannot be found, the allocator rejects the request, sets the appropriate flag, and returns.

Once it is determined that the request can be satisfied, the core is allocated to that task and the core map is updated.

C. Dispatcher

The Dispatcher is the routine through which EXEC 8 accomplishes its time-sharing processes. It allows CPU's to be assigned to and released from specific task execution as various events and contingencies arise (e.g., I/O-initiations). CPU priority is computed on the basis of task switch list level (see Figure 8).

A task will lose control of a CPU in favor of another task at the currently lowest switch list level either upon expiration of a set time quantum, completion of the task's CPU requirements, or by voluntarily releasing control while awaiting completion of an I/O request.

The CPU time quantum has a value of 8 milliseconds for level 1, and is doubled for each subsequent level up to a maximum of 1.024 seconds at level 8.

Batch-type runs enter the list at level 2.

The Dispatcher module in the simulator performs a triple function in determining:

1) which CPU will be allocated to a particular task,

2) which of three events will terminate the task's current tenure on the CPU, and

3) the duration of the time slice.

The three events that can terminate a time slice are:

(a) completion of a time quantum,

(b) initiation of an I/O request by the task, or

(c) completion of the task's CPU time requirements.

If (a) is the case, the task will utilize the assigned CPU for the appropriate time quantum, and then reenter the CPU queue at a switch list level one greater than before (up to a maximum level of 8). In the case of event (b) the task will remain in execution until the I/O initiation and then enter a statistically determined I/O subsystem queue. Upon access to the subsystem the task utilizes it for a statistically determined amount of service time, and then reenters the CPU queue at the lowest switch list level for which it is eligible. In the case of event (c), the task will run out its remaining time on the selected CPU, release the core associated with it, and attach the next task of the run to the core queue. When the last task of a run completes, the facilities for that run are released and the run terminates.

The Dispatcher logic flow is illustrated in Figure 9.

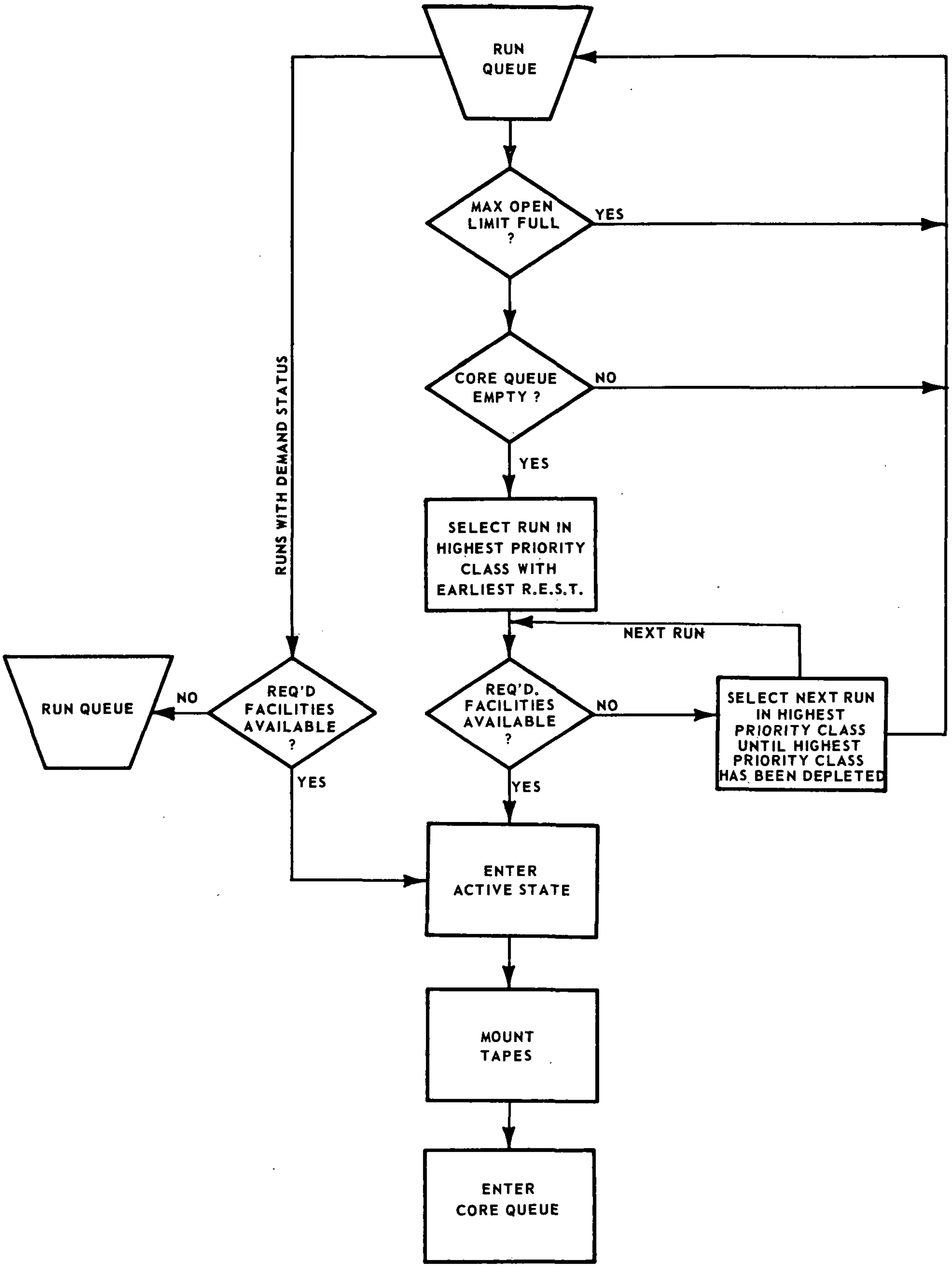


FIGURE 6. COARSE SCHEDULER– LOGIC FLOW

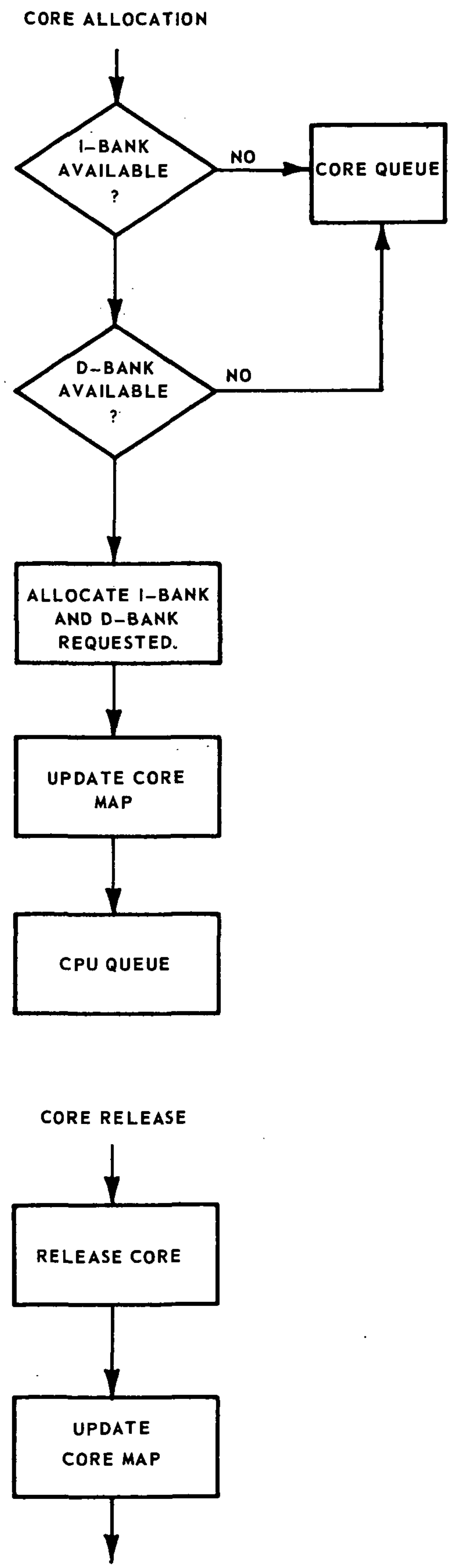


FIGURE 7. DYNAMIC ALLOCATOR – LOGIC FLOW

LEVEL	TIME QUANTUM (MILLISECONDS)
1	8
2	16
3	32
4	64
5	128
6	256
7	512
8	1024

FIGURE 8 DISPATCHER SWITCH LIST

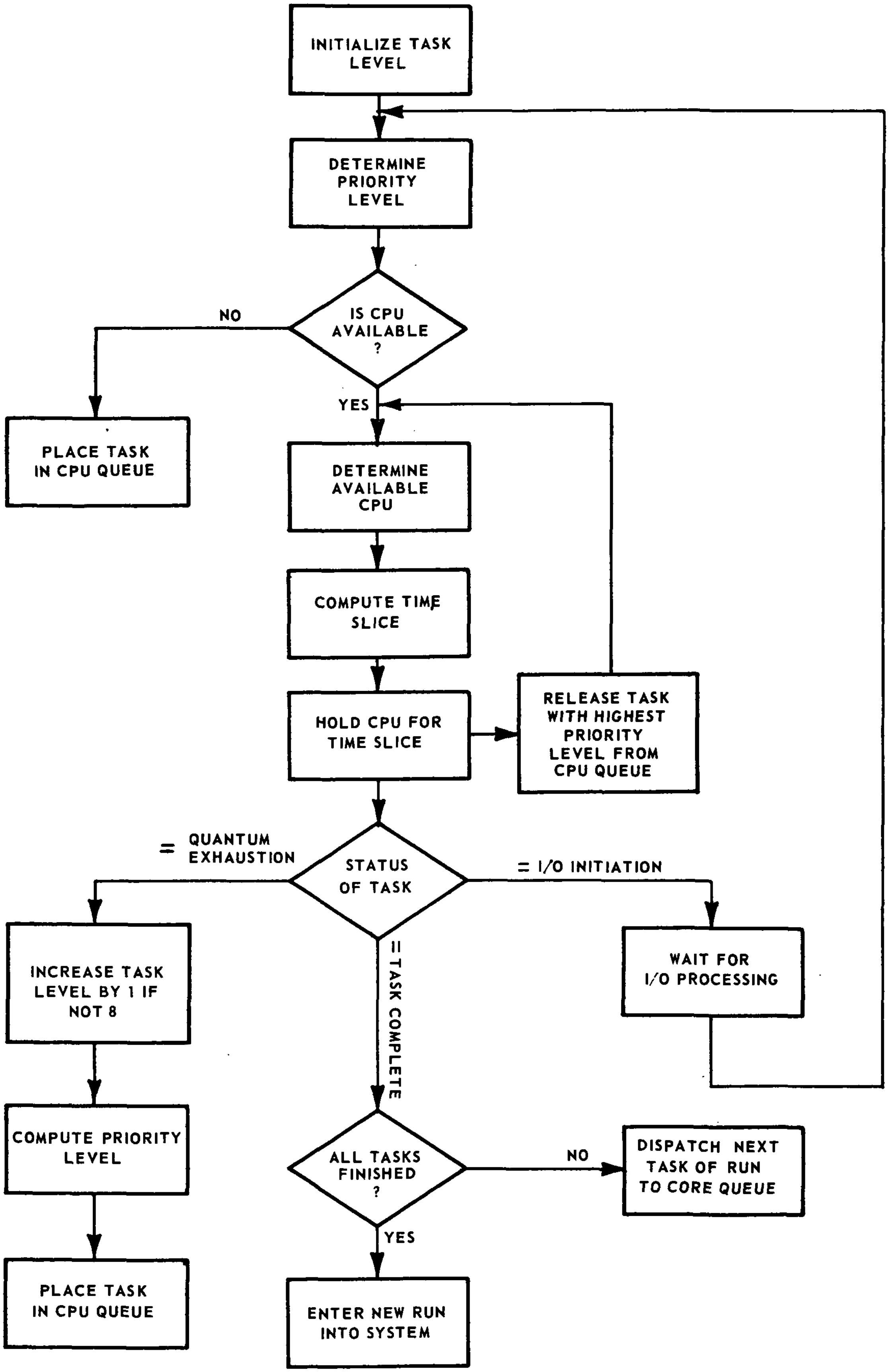


FIGURE 9. DISPATCHER –LOGIC FLOW

SECTION IV. MODEL VERIFICATION

A. Initial Verification

In order to verify the simulator it was considered necessary to compare simulation results with known actual results.

Hence, the accounting tape associated with a specific first shift period was used to produce a jobtape as input to the model. This tape consisted of approximately 150 runs and represented almost two hours of system elapsed time.

For the purpose of verifying the simulator throughput performance, the following method is used.

A value, V_r, is maintained for each run, r, which has entered the system (i.e., had its run card read). If the run has not terminated,

$$V_r = (\text{Current Time}) - (\text{R.E.S.T.})$$

If the run has terminated,

$$V_r = (\text{Run Termination Time}) - (\text{R.E.S.T.})$$

The quantity,

$$S = \left(\sum_{r=1}^{\text{NOESYS}} V_r \right) / \text{NOESYS},$$

where NOESYS is the number of runs which have entered the system, is computed and stored at the end of every simulated minute.

Figure 10 shows the system throughput performance as measured for the verification of the simulator.

B. Further Verification

Because of certain assumptions which have been made with regard to such statistics areas as I/O blocking factor, I/O initiation pattern of EXEC tasks, task I/O subsystem distribution, and tape mount time, a more controlled verification is desired. Currently in progress is an attempt at this verification. A benchmark runstream representative of an actual MSFC U-1108/8 runstream is being constructed. This synthetic runstream will have the same statistical qualities as the actual runstream selected. (On a run basis: the I/O subsystem distribution for physical I/O's, number of tape drives, and number of lines output; and on a task basis: I-bank/D-bank core requirements, number of physical I/O's, and CPU time requirements.) Using this synthetic runstream, two experiments will be performed on the captive MSFC U-1108/8:

1. Normal System

2. Degraded System (one FASTRAND controller will be disabled)

In each case a simulator jobtape will be constructed from the system accounting tape statistics accumulated during the experiment and the experiment simulated. Successful duplication of results will then constitute the sought after verification.

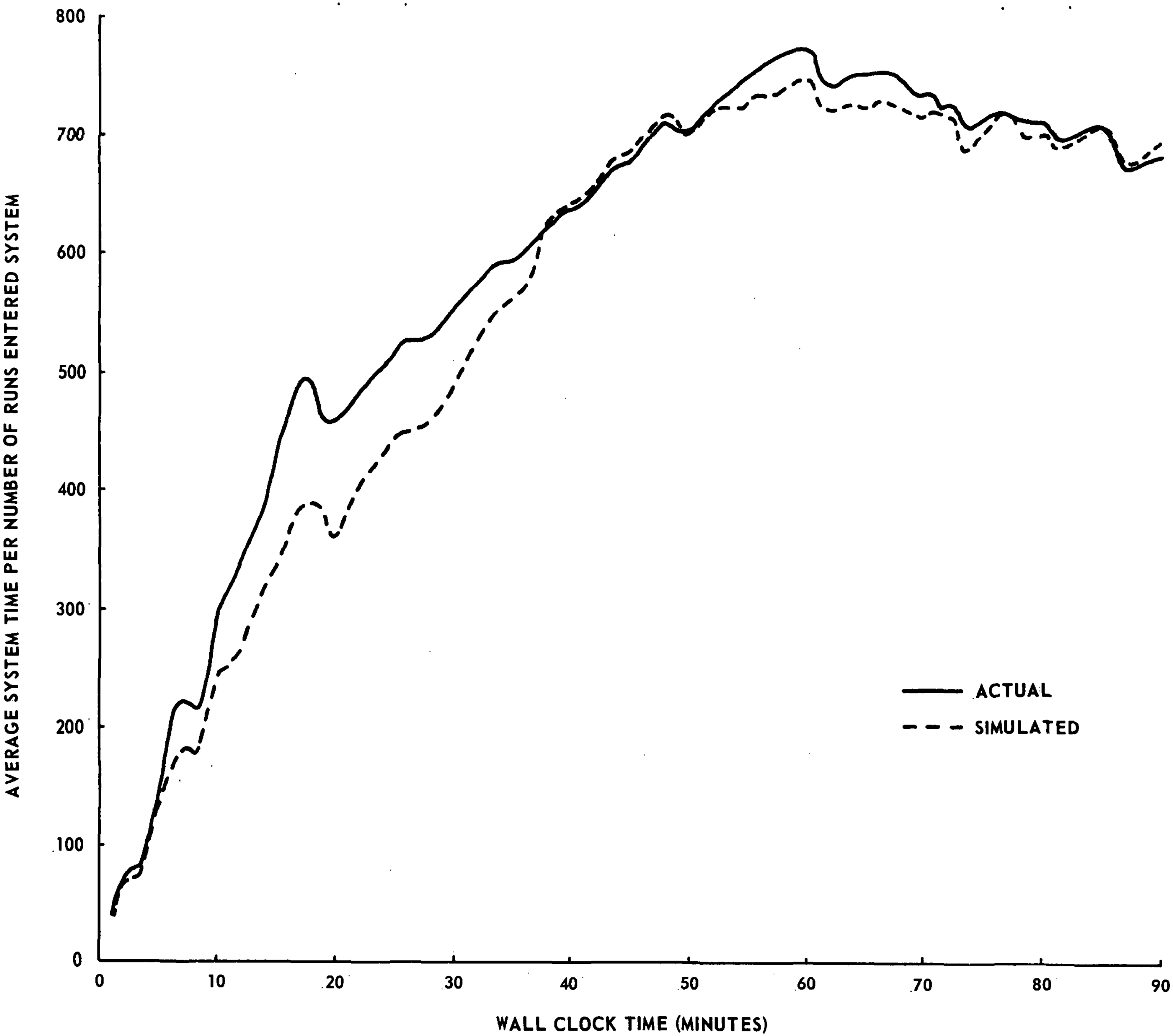


FIGURE 10. MODEL VERIFICATION

SECTION V. SIMULATOR APPLICATIONS

A. Introduction

The simulator has been used to analyze the performance of the MSFC UNIVAC 1108. This was done by performing a series of tests which (a) systematically perturbed each part of the Executive System whose function is the allocation of the various 1108 resources, and (b) varied the existing hardware configuration. Resource allocation is performed by three separate parts of the Executive System, viz., the Coarse Scheduler, the Dynamic Allocator and the Dispatcher which are responsible for the allocation of tapes, core and CPU time, respectively.

The results of tests performed on the functions of the Coarse Scheduler and Dynamic Allocator pointed to the inordinately long delays being encountered by I/O requests to the FASTRAND drums as the major factor limiting system throughput. Hence, another series of tests was initiated which investigated the effect of (a) increasing the number of FASTRAND dual channel controllers from two to three, and (b) replacing the FASTRAND drums by 8440 type disks while maintaining the same volume of mass storage. This set of experiments showed that either of the above strategies would relieve the I/O channel bottleneck. In order to investigate the sensitivity of the system throughput to perturbation of the Dispatcher algorithm, all experiments with CPU time slicing were carried out on a system having three FASTRAND dual-channel controllers and whose core allocation was optimized by compaction. Selected experiments were re-run on a third-shift job mix.

The complete set of experiments performed is shown in Table 1.

Experiments			Results Summary	
Shift	No.	Description	Table	Figure
1	1	Baseline Configuration	2	12
1	2	Zero Tape-Mount Time	3	12
1	3	Tapes Mounted Before Run Opened	4	12
1	4	MAXOPEN = 20	5	12
1	5	Assiging Core Nearest Edge	6	13
1	6	Flip-Flop Core Assignment Algorithm	7	13
1	7	Core Compaction	8	13
1	8	Core Compaction with MAXOPEN = 20	9	13
1	9	Re-Entrant FORTRAN Compiler	10	13
1	10	Three FASTRAND Controllers	11	14
1	11	Two 8440 Disk Controllers	12	14
1	12	Three 8440 Disk Controllers	13	14

Table 1. Experiments Summary

Experiment			Results Summary	
Shift	No.	Description	Table	Figure
1	13	Core Compaction and Three FASTRAND Controllers	14	15
1	14	Core Compaction, Three FASTRAND Controllers and MAXOPEN = 20	15	15
1	15	Core Compaction and Two 8440 Disk Controllers	16	15
1	16	Core Compaction and Three 8440 Disk Controllers	17	15
1	17	Core Compaction, Three FASTRAND Controllers and Double-Time Quantum	18	16
1	18	Core Compaction, Three FASTRAND Controllers and Half-Time Quantum	19	16
1	19	Core Compaction, Three FASTRAND Controllers and Time Quantum Termination on an I/O Operation	20	16
1	20	Core Compaction, Three FASTRAND Controllers and CPU Priority as a Function of I/O Frequency	21	16
3	21	Baseline Configuration	22	17
3	22	Zero Tape-Mount Time	23	17
3	23	Core Compaction	24	17
3	24	Three FASTRAND Controllers	25	17

Table 1 (cont.). Experiments Summary

The results of all the experiments are summarized in Tables 2 through 25 and Figures 11 through 17 which show the amounts of total system time spent by the work load components queueing for the use of system resources (tapes, core, CPU's, and I/O channels) together with the percentage utilization of core and CPU's. The net effect on system throughput is represented by the change in total elapsed time taken by the system to process the fixed work load. Experiment #1 represents the 1108 baseline configuration as verified in Figure 10. The results of the baseline performance are given in each of Figures 12 through 16 for ease of comparison with all other simulator predictions.

All experiments were performed using a common job mix for each shift. The first shift mix consisted of 150 runs selected at random from a total of approximately 1,000 runs which had been extracted from five different accounting tapes produced over a five-day period. The third shift mix was made up of 60 runs randomly selected from a total of approximately 800 runs taken from five accounting tapes which had been recorded over a five-day period.

The run arrival rate was determined by the MAXOPEN parameter, in that the total work load was entered into the simulated run queue and

runs were extracted from there by the system under the constraint that the MAXOPEN limit (held constant at 10) was not exceeded.

The elapsed time given in Tables 2 through 25 refers to the time which elapses between the total work load entering the simulated run queue and the last run terminating.

Results Summary for Experiment # 1 BASELINE CONFIGURATION - FIRST SHIFT	
Total Elapsed Time (min:sec)	100:54
Time in Tape Mount Queue (sec)	3995
Time in Core Queue (sec)	7896
Time in CPU Queue (sec)	1382
Time in Channel Queues (sec)	30,049
CPU Utilization (%)	52.76
Core Utilization (%)	75.00

Table 2 - Results Summary For Experiment #1

Results Summary for Experiment # 2 ZERO TAPE MOUNT TIME - FIRST SHIFT	
Total Elapsed Time (min:sec)	101:44
Time in Tape Mount Queue (sec)	0
Time in Core Queue (sec)	8729
Time in CPU Queue (sec)	1774
Time in Channel Queues (sec)	35,526
CPU Utilization (%)	52.50
Core Utilization (%)	77.13

Table 3 - Results Summary for Experiment #2

Results Summary for Experiment # 3 TAPES MOUNTED BEFORE RUN OPENED - FIRST SHIFT	
Total Elapsed Time (min:sec)	100.57
Time in Tape Mount Queue (sec)	6,420
Time in Core Queue (sec)	8903
Time in CPU Queue (sec)	1323
Time in Channel Queues (sec)	33,344
CPU Utilization (%)	53.01
Core Utilization (%)	76.63

Table 4 - Results Summary For Experiment #3

Results Summary for Experiment # 4 MAXOPEN EQUALS 20 - FIRST SHIFT	
Total Elapsed Time (min:sec)	98.56
Time in Tape Mount Queue (sec)	6055
Time in Core Queue (sec)	13,059
Time in CPU Queue (sec)	1490
Time in Channel Queues (sec)	33,501
CPU Utilization (%)	53.83
Core Utilization (%)	78.30

Table 5 - Results Summary for Experiment #4

Results Summary for Experiment # 5 ASSIGNING CORE NEAREST EDGE - FIRST SHIFT	
Total Elapsed Time (min:sec)	100:03
Time in Tape Mount Queue (sec)	3482
Time in Core Queue (sec)	7804
Time in CPU Queue (sec)	1459
Time in Channel Queues (sec)	34,849
CPU Utilization (%)	53.36
Core Utilization (%)	77.33

Table 6 - Results Summary For Experiment #5

Results Summary for Experiment # 6 FLIP-FLOP CORE ASSIGNMENT ALGORITHM - FIRST SHIFT	
Total Elapsed Time (min:sec)	98:10
Time in Tape Mount Queue (sec)	3240
Time in Core Queue (sec)	7657
Time in CPU Queue (sec)	1588
Time in Channel Queues (sec)	40,500
CPU Utilization (%)	54.30
Core Utilization (%)	82.12

Table 7 - Results Summary for Experiment #6

Results Summary for Experiment # 7 CORE COMPACTION - FIRST SHIFT	
Total Elapsed Time (min:sec)	96.51
Time in Tape Mount Queue (sec)	2731
Time in Core Queue (sec)	6102
Time in CPU Queue (sec)	1877
Time in Channel Queues (sec)	49,200
CPU Utilization (%)	54.97
Core Utilization (%)	87.86

Table 8 - Results Summary For Experiment #7

Results Summary for Experiment # 8 CORE COMPACTION AND MAXOPEN EQUALS 20 - FIRST SHIFT	
Total Elapsed Time (min:sec)	96:7
Time in Tape Mount Queue (sec)	7036
Time in Core Queue (sec)	10726
Time in CPU Queue (sec)	1881
Time in Channel Queues (sec)	53,761
CPU Utilization (%)	55.47
Core Utilization (%)	90.79

Table 9 - Results Summary for Experiment #8

Results Summary for Experiment # 9 RE-ENTRANT FORTRAN COMPILER - FIRST SHIFT	
Total Elapsed Time (min:sec)	104:00
Time in Tape Mount Queue (sec)	3931
Time in Core Queue (sec)	10,421
Time in CPU Queue (sec)	1388
Time in Channel Queues (sec)	29,713
CPU Utilization (%)	51.24
Core Utilization (%)	74.08

Table 10- Results Summary For Experiment #9

Results Summary for Experiment # 10 THREE FASTRAND CONTROLLERS - FIRST SHIFT	
Total Elapsed Time (min:sec)	81:55
Time in Tape Mount Queue (sec)	3981
Time in Core Queue (sec)	6586
Time in CPU Queue (sec)	2488
Time in Channel Queues (sec)	13,893
CPU Utilization (%)	65.03
Core Utilization (%)	77.62

Table 11- Results Summary for Experiment #10

Results Summary for Experiment # 11 TWO 8440 DISK CONTROLLERS - FIRST SHIFT	
Total Elapsed Time (min:sec)	80:37
Time in Tape Mount Queue (sec)	4444
Time in Core Queue (sec)	6530
Time in CPU Queue (sec)	2530
Time in Channel Queues (sec)	9535
CPU Utilization (%)	66.56
Core Utilization (%)	70.45

Table 12- Results Summary For Experiment #11

Results Summary for Experiment # 12 THREE 8440 DISK CONTROLLERS - FIRST SHIFT	
Total Elapsed Time (min:sec)	73:52
Time in Tape Mount Queue (sec)	4560
Time in Core Queue (sec)	6300
Time in CPU Queue (sec)	3842
Time in Channel Queues (sec)	4688
CPU Utilization (%)	72.48
Core Utilization (%)	73.10

Table 13- Results Summary for Experiment #12

Results Summary for Experiment # 13 CORE COMPACTION AND THREE FASTRAND CONTROLLERS - FIRST SHIFT	
Total Elapsed Time (min:sec)	76:22
Time in Tape Mount Queue (sec)	3588
Time in Core Queue (sec)	4011
Time in CPU Queue (sec)	3734
Time in Channel Queues (sec)	20,652
CPU Utilization (%)	70.00
Core Utilization (%)	86.86

Table 14- Results Summary For Experiment #13

Results Summary for Experiment # 14 CORE COMPACTION, THREE FASTRAND CONTROLLERS AND MAXOPEN EQUALS 20 - FIRST SHIFT	
Total Elapsed Time (min:sec)	74:17
Time in Tape Mount Queue (sec)	10,633
Time in Core Queue (sec)	7043
Time in CPU Queue (sec)	4966
Time in Channel Queues (sec)	21,768
CPU Utilization (%)	71.98
Core Utilization (%)	90.19

Table 15- Results Summary for Experiment #14

Results Summary for Experiment # 15 CORE COMPACTION AND TWO 8440 DISK CONTROLLERS - FIRST SHIFT	
Total Elapsed Time (min:sec)	69:36
Time in Tape Mount Queue (sec)	3783
Time in Core Queue (sec)	4119
Time in CPU Queue (sec)	4727
Time in Channel Queues (sec)	16,422
CPU Utilization (%)	76.71
Core Utilization (%)	85.81

Table 16 - Results Summary For Experiment #15

Results Summary for Experiment # 16 CORE COMPACTION AND THREE 8440 DISK CONTROLLERS - FIRST SHIFT	
Total Elapsed Time (min:sec)	65:00
Time in Tape Mount Queue (sec)	4494
Time in Core Queue (sec)	3631
Time in CPU Queue (sec)	6066
Time in Channel Queues (sec)	7338
CPU Utilization (%)	82.31
Core Utilization (%)	83.94

Table 17 - Results Summary for Experiment #16

Results Summary for Experiment # 17 CORE COMPACTION, THREE FASTRAND CONTROLLERS AND DOUBLE TIME QUANTUM - FIRST SHIFT	
Total Elapsed Time (min:sec)	76:56
Time in Tape Mount Queue (sec)	4223
Time in Core Queue (sec)	3799
Time in CPU Queue (sec)	2590
Time in Channel Queues (sec)	19,496
CPU Utilization (%)	69.99
Core Utilization (%)	85.96

Table 18 - Results Summary For Experiment #17

Results Summary for Experiment # 18 CORE COMPACTION, THREE FASTRAND CONTROLLERS AND HALF TIME QUANTUM - FIRST SHIFT	
Total Elapsed Time (min:sec)	75.59
Time in Tape Mount Queue (sec)	3444
Time in Core Queue (sec)	4055
Time in CPU Queue (sec)	2906
Time in Channel Queues (sec)	19,753
CPU Utilization (%)	70.46
Core Utilization (%)	87.04

Table 19 - Results Summary for Experiment #18

Results Summary for Experiment # 19 CORE COMPACTION, THREE FASTRAND CONTROLLERS AND TIME QUANTUM TERMINATING ON AN I/O OPERATION - FIRST SHIFT	
Total Elapsed Time (min:sec)	76:39
Time in Tape Mount Queue (sec)	3800
Time in Core Queue (sec)	4239
Time in CPU Queue (sec)	5108
Time in Channel Queues (sec)	21,718
CPU Utilization (%)	69.82
Core Utilization (%)	86.18

Table 20 - Results Summary For Experiment #19

Results Summary for Experiment # 20 CORE COMPACTION, THREE FASTRAND CONTROLLERS AND CPU PRIORITY AS A FUNCTION OF I/O FREQUENCY - FIRST SHIFT	
Total Elapsed Time (min:sec)	75:46
Time in Tape Mount Queue (sec)	3588
Time in Core Queue (sec)	3875
Time in CPU Queue (sec)	5881
Time in Channel Queues (sec)	23,210
CPU Utilization (%)	70.45
Core Utilization (%)	86.60

Table 21 - Results Summary for Experiment #20

Results Summary for Experiment # 21 BASELINE CONFIGURATION - THIRD SHIFT	
Total Elapsed Time (min:sec)	125:40
Time in Tape Mount Queue (sec)	2413
Time in Core Queue (sec)	7464
Time in CPU Queue (sec)	1119
Time in Channel Queues (sec)	29,471
CPU Utilization (%)	50.95
Core Utilization (%)	70.48

Table 22 - Results Summary For Experiment #21

Results Summary for Experiment # 22 ZERO TAPE MOUNT TIME - THIRD SHIFT	
Total Elapsed Time (min:sec)	127:9
Time in Tape Mount Queue (sec)	0
Time in Core Queue (sec)	9689
Time in CPU Queue (sec)	959
Time in Channel Queues (sec)	28,628
CPU Utilization (%)	50.39
Core Utilization (%)	69.54

Table 23 - Results Summary for Experiment #22

Results Summary for Experiment # 23 CORE COMPACTION - THIRD SHIFT	
Total Elapsed Time (min:sec)	121:9
Time in Tape Mount Queue (sec)	1963
Time in Core Queue (sec)	5015
Time in CPU Queue (sec)	1615
Time in Channel Queues (sec)	39,620
CPU Utilization (%)	53.00
Core Utilization (%)	77.64

Table 24- Results Summary For Experiment #23

Results Summary for Experiment # 24 THREE FASTRAND CONTROLLERS - THIRD SHIFT	
Total Elapsed Time (min:sec)	109:30
Time in Tape Mount Queue (sec)	2,827
Time in Core Queue (sec)	9532
Time in CPU Queue (sec)	3028
Time in Channel Queues (sec)	9264
CPU Utilization (%)	58.46
Core Utilization (%)	69.41

Table 25- Results Summary for Experiment #24

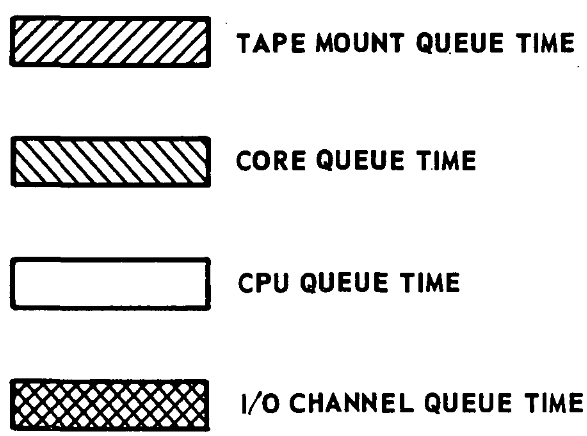


FIGURE 11. KEY TO FIGURES 12–17

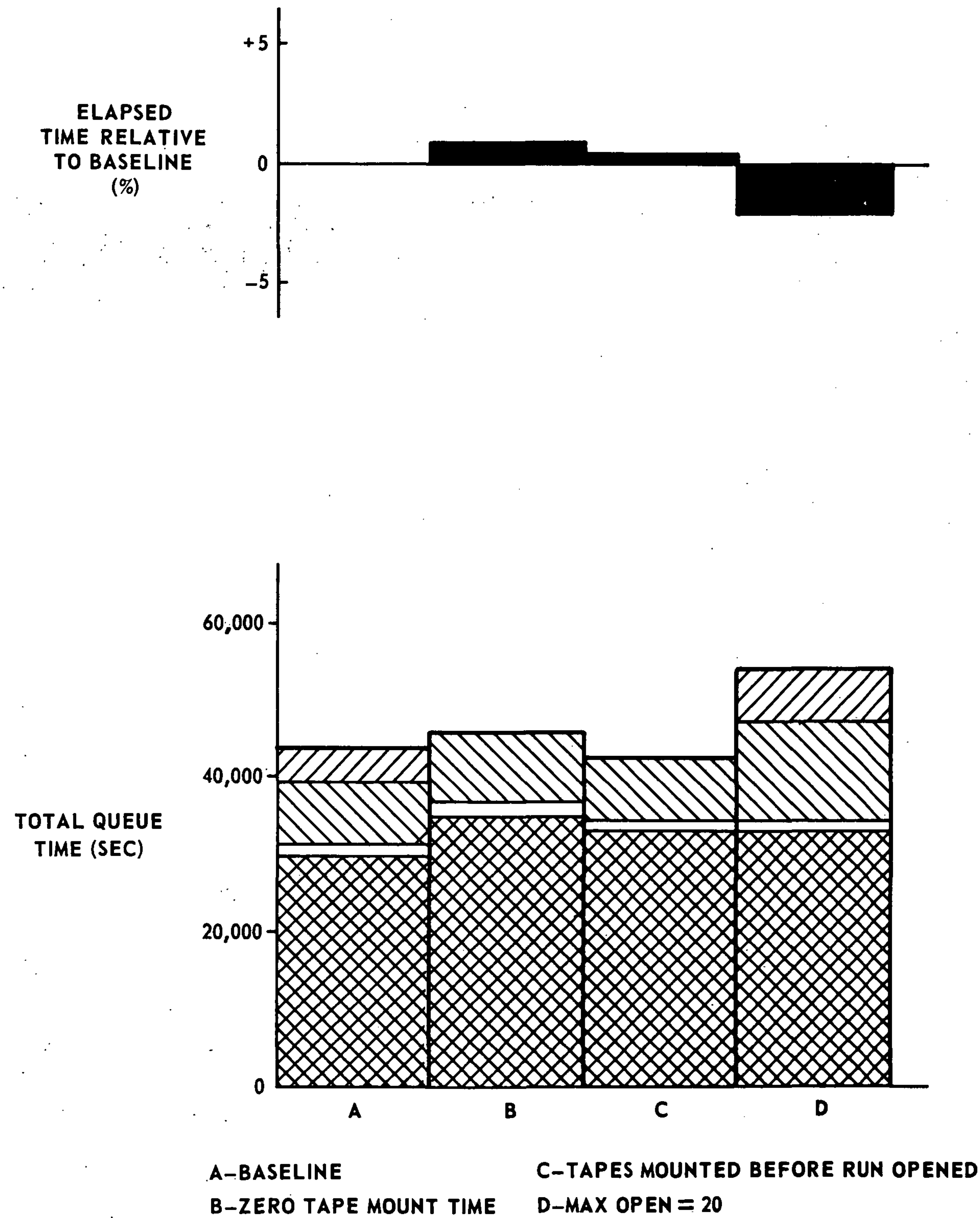
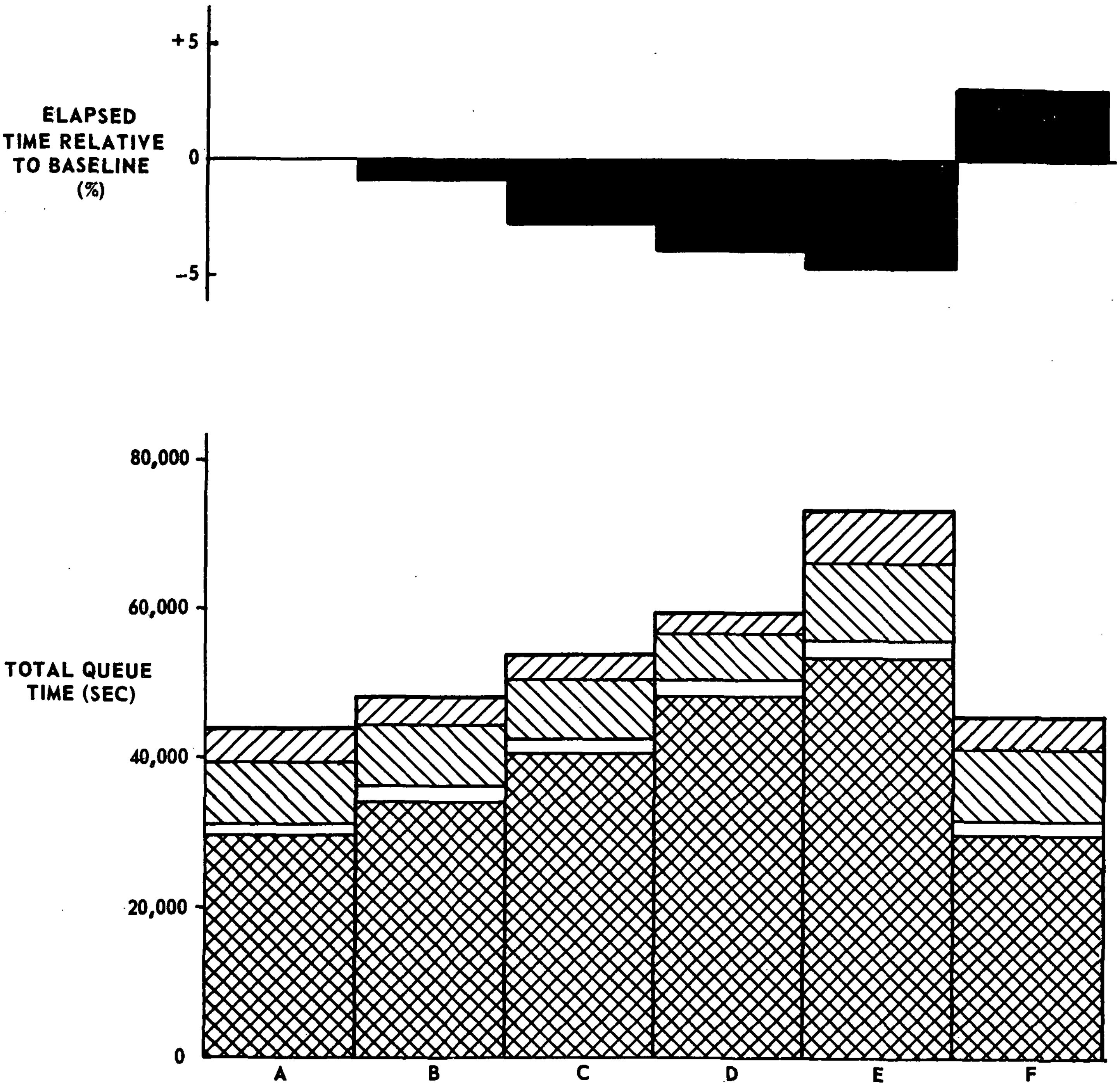
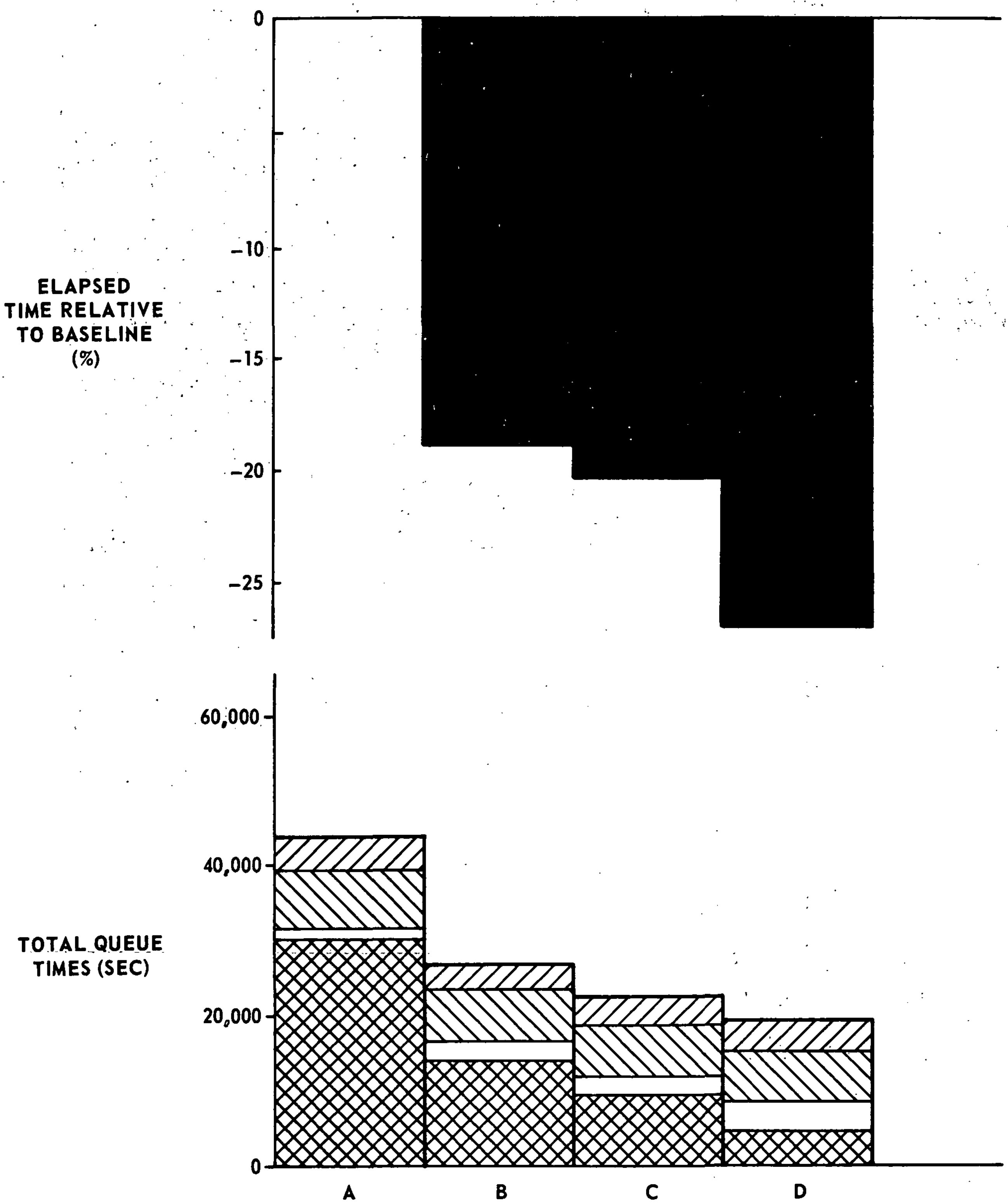


FIGURE 12. TAPE AND SYSTEM RELATED EXPERIMENTS



A – BASELINE
B – ASSIGNING CORE NEAREST EDGE
C – FLIP-FLOP CORE ASSIGNMENT ALGORITHM
D – CORE COMPACTION
E – CORE COMPACTION WITH MAX OPEN = 20
F – RE-ENTRANT FORTRAN COMPILER

FIGURE 13. CORE RELATED EXPERIMENTS



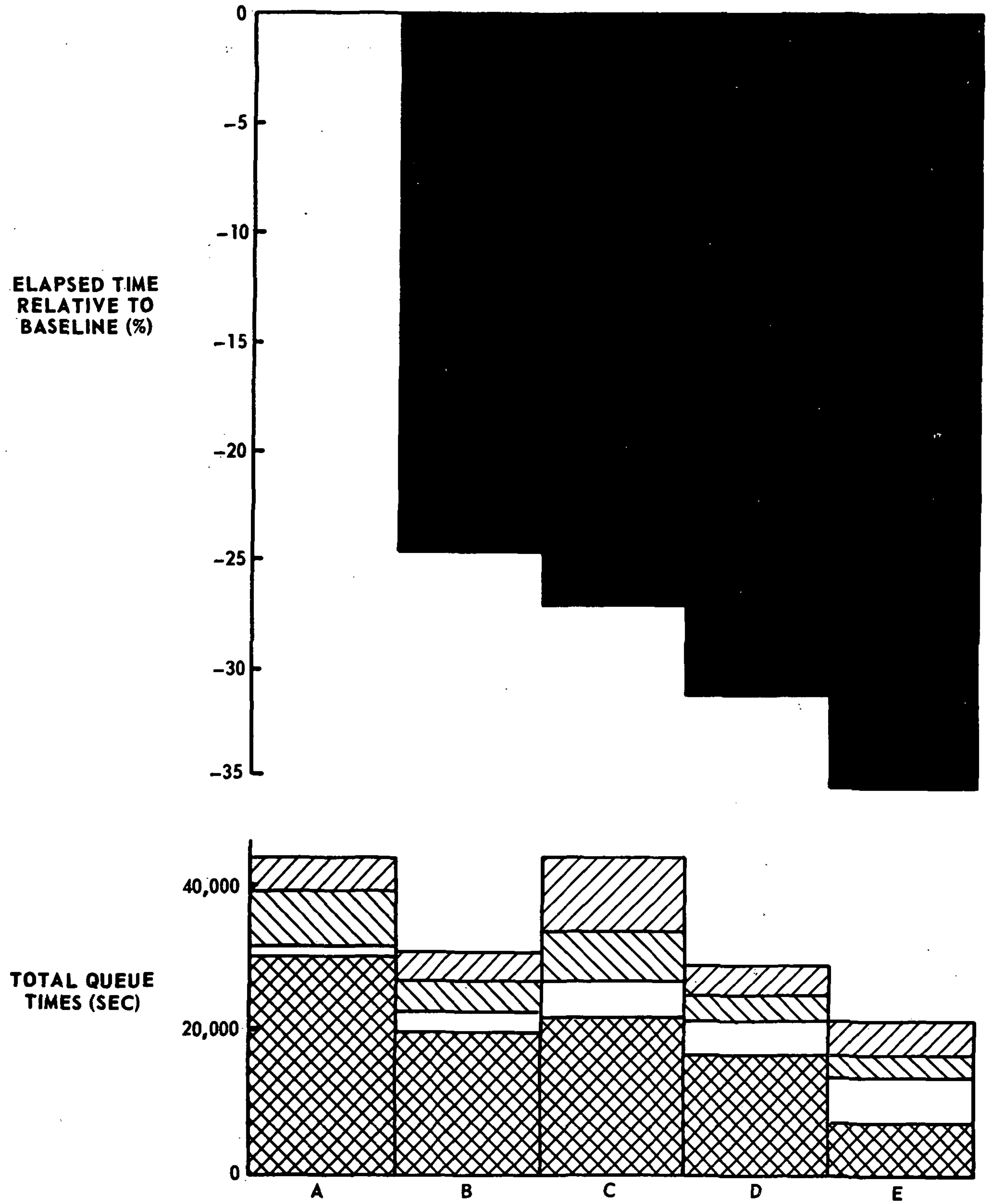
A – BASELINE

B – 3 FR CONTROLLERS

C – 2,8440 DISK CONTROLLERS

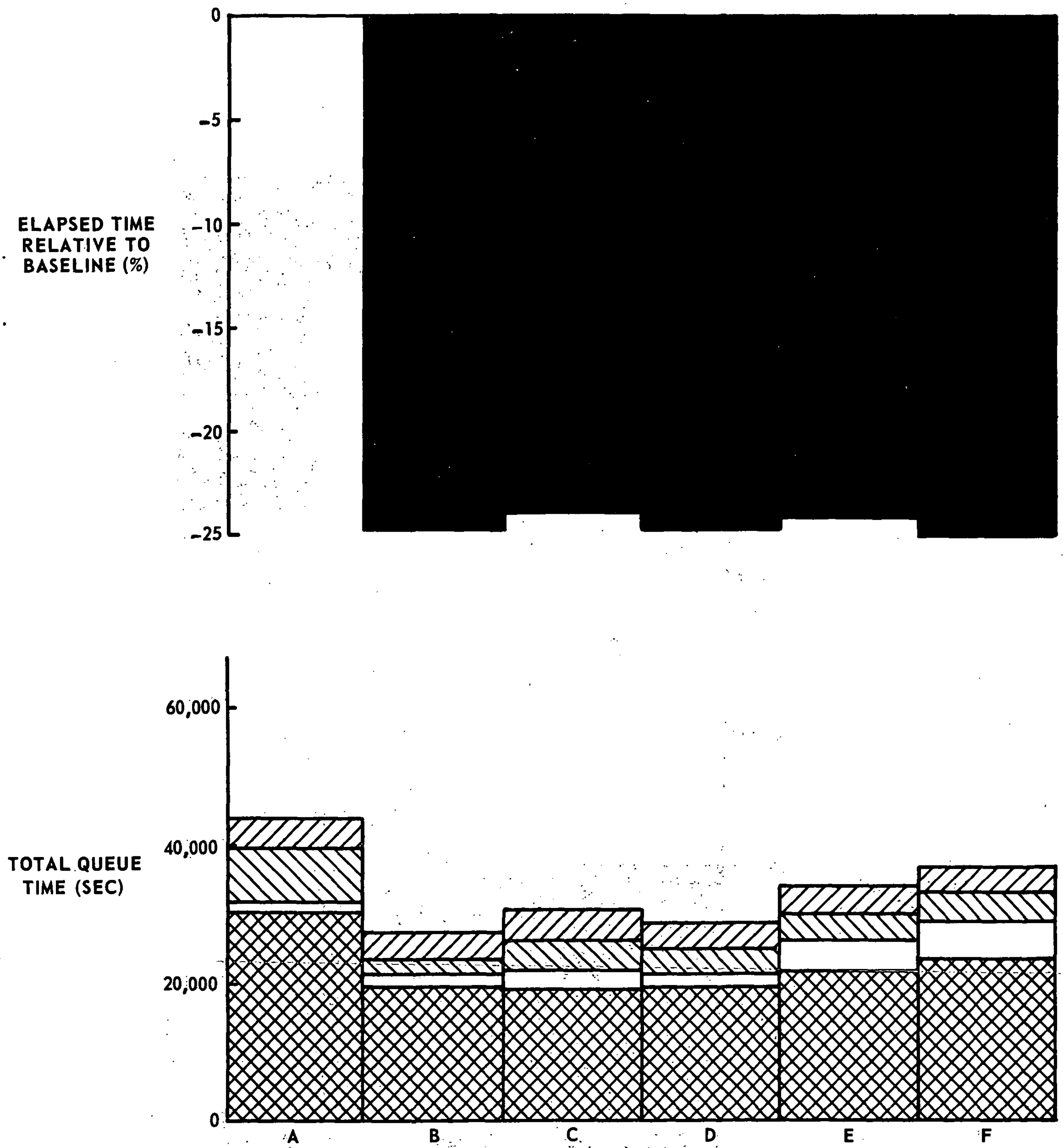
D – 3,8440 DISK CONTROLLERS

FIGURE 14. I/O RELATED EXPERIMENTS



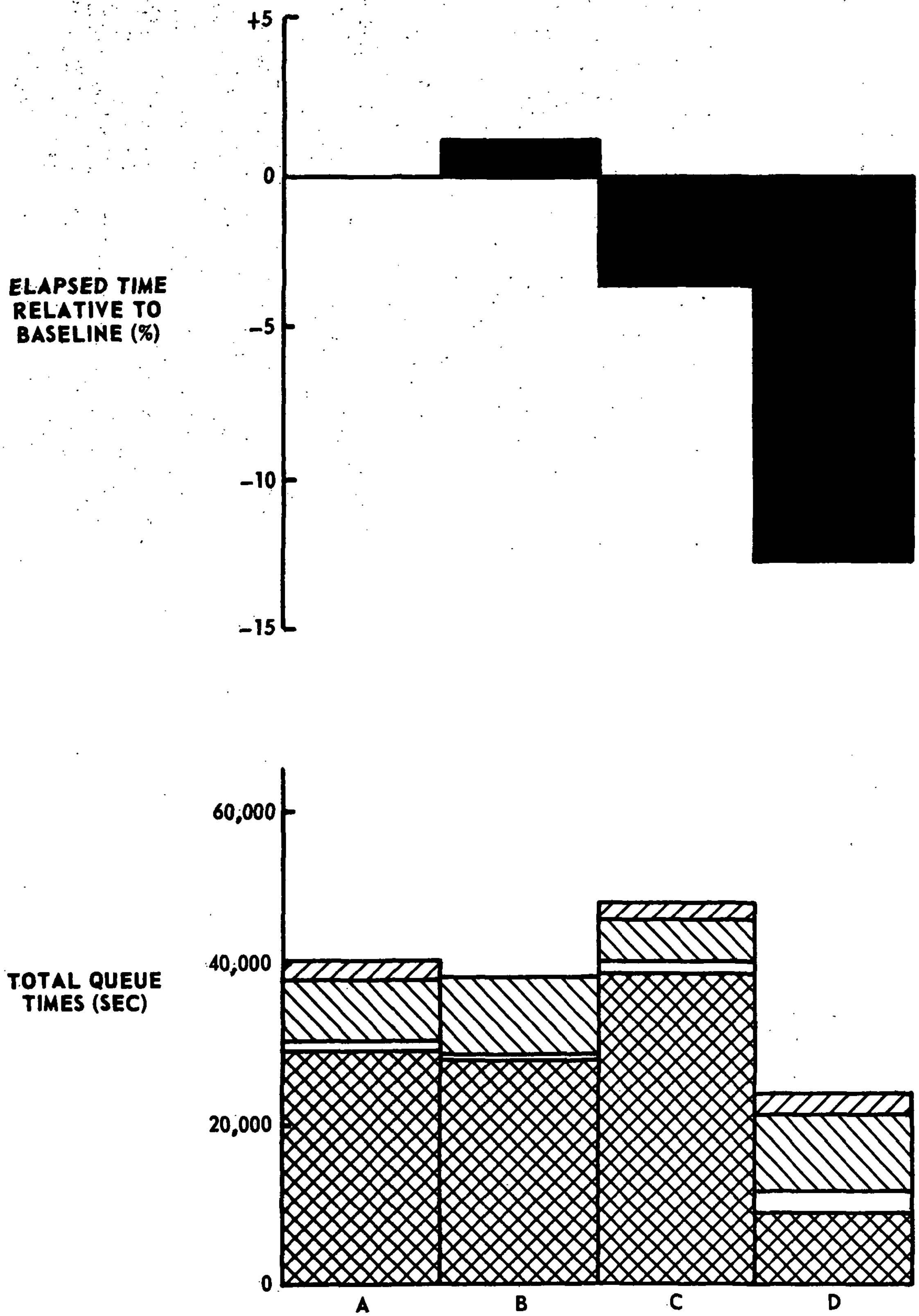
A - BASELINE
B - CORE COMPACTION AND 3 FR CONTROLLERS
C - CORE COMPACTION, 3 FR CONTROLLERS AND MAX OPEN = 20
D - CORE COMPACTION AND 2,8440 DISK CONTROLLERS
E - CORE COMPACTION AND 3,8440 DISK CONTROLLERS

FIGURE 15. THROUGHPUT MAXIMIZATION EXPERIMENTS



A – BASELINE
B – CORE COMPACTION WITH 3 FR CONTROLLERS
C – CORE COMPACTION, 3 FR CONTROLLERS AND DOUBLE TIME QUANTUM
D – CORE COMPACTION, 3 FR CONTROLLERS AND HALF TIME QUANTUM
E – CORE COMPACTION, 3 FR CONTROLLERS AND TIME QUANTUM TERMINATING ON I/O
F – CORE COMPACTION, 3 FR CONTROLLERS AND CPU PRIORITY AS FUNCTION OF I/O FREQ.

FIGURE 16. CPU TIME QUANTUM SENSITIVITY EXPERIMENTS



A – BASELINE
B – ZERO TAPE MOUNT TIME
C – CORE COMPACTION
D – 3 FR CONTROLLERS

FIGURE 17. THIRD SHIFT EXPERIMENTS

B. Discussion of Results

TAPE AND SYSTEM RELATED EXPERIMENTS

<u>Experiment #1:</u> Baseline Configuration - First Shift (Ref: Table 2 and Figure 12)

The total queueing time in the baseline was 43,321 sec; this compares to a total CPU time of 9,582 sec.

<u>Experiment #2:</u> Zero Tape Mount Time (Ref: Table 3 and Figure 12)

This experiment simulated the ultimate in efficiency in mounting tapes, viz., zero time spent in the actual mounting procedure. It can be seen from Figure 12 that although the time spent in the tape mount queue was eliminated, the overall processing time increased because of the limiting effect of the increased time spent in the channel queues.

<u>Experiment #3:</u> Tapes Mounted Before Run Opened (Ref: Table 4 and Figure 12)

In the operation of the baseline EXEC 8 operating system, when a run is opened it contributes to the MAXOPEN count. However, if such a run has a requirement for one or more tape drives, time is required to locate and mount the necessary tapes. It can be argued that in such a situation the total system is not being fully utilized because until a run has its tapes mounted it cannot vie for other system resources such as core and CPU time.

The above experiment simulates the effect of premounting a run's tapes before the run is opened. In this case when a run enters the system by virtue of satisfying the MAXOPEN criterion, it can immediately become a candidate for core allocation. This procedure virtually eliminates the effect of the time spent waiting in the tape mount queue since this activity occurs in parallel with the processing of core-resident tasks under conditions of a fully loaded system with respect to the MAXOPEN criterion.

Although core and CPU utilization increased marginally, throughput was almost unaffected due to an increase in I/O channel queue time.

<u>Experiment #4:</u> MAXOPEN = 20 (Ref: Table 5 and Figure 12)

The MAXOPEN limit was raised from 10 to 20, reducing the total elapsed time by almost 2 percent. The degree of improvement was limited, however, by an increase in both the core and I/O channel queue times.

CORE RELATED EXPERIMENTS

Experiment #5: Assigning Core Nearest Edge (Ref: Table 6 and Figure 13)

In this experiment the "best fit" algorithm normally used for placing a task in core is replaced by an algorithm which attempts to keep the I-bank and D-bank portions of the task separated as much as possible. A task's I-bank is placed in the core gap with smallest beginning address which is large enough to contain it; task D-bank is placed in the appropriate gap with largest beginning address. This algorithm attempts to preserve a large center gap so that a task having a large core requirement can be quickly accommodated.

Only a marginal improvement in throughput was noted, again accompanied by an increase in I/O channel queue time.

Experiment #6: Flip-Flop Core Assignment Algorithm (Ref: Table 7 and Figure 13)

From the baseline core allocation algorithm described in Section III.B., it can be seen that I-bank is always allocated below and D-bank always above the center gap. This experiment investigates the effect of modifying this algorithm such that the only requirement is that I- and D-bank be on opposite sides of the center gap. This results in a significant increase in core utilization, but the increase in throughput is less than 3 percent, again due mainly to the increase in I/O channel queueing time.

Experiment #7: Core Compaction (Ref: Table 8 and Figure 13)

In a multiprogramming environment unused core is typically fragmented into several separate blocks or gaps. This fragmentation can be substantially reduced by compacting core; that is, relocating tasks so as to reduce the number of blocks into which unused core is divided. In this experiment the optimum core compaction algorithm is assumed; when a core request is made, those tasks residing in core are relocated so that all unused core is contained in one gap. This allows utilization of the unused fragments which occur when any of the above algorithms are employed. The overall effect is an increased throughput of over 4 percent owing to an increase in core utilization from 75 percent in the baseline configuration to 87.86 percent. However, the increased user activity causes more rapid requests for I/O thereby creating a large increase in I/O channel queueing times.

Experiment #8: Core Compaction and MAXOPEN = 20 (Ref: Table 9 and Figure 13)

This experiment achieves a core utilization of 90.79 percentm but an increase in channel queueing time of almost 80 percent over the baseline figure restricts the net reduction in processing time to less than 5 percent.

Experiment #9: Re-entrant Fortran Compiler (Ref: Table 10 and Figure 13)

The effect of a re-entrant Fortran compiler was simulated by allowing any concurrent Fortran compile tasks to execute from a common 16.38K I-bank. No improvement in throughput was noted. However, core utilization decreased suggesting that the baseline core allocation algorithm with a floating center gap was not compatible with the re-entrant compiler. Further investigation is needed in this area.

I/O RELATED EXPERIMENTS

From the results discussed so far, it is obvious that a bottleneck exists on the I/O channels. Study of the individual channel queuing statistics points to the Fastrand subsystems as the location where the longest queue times are experienced. The following set of experiments describes the effects of increasing the number of Fastrand dual controllers from two to three and also of replacing all Fastrand by a type 8440 disk system.

An I/O service time of 70 msec, derived from the Univac THRPUT program, was assumed for all Fastrand channels. However, since no such figure was available for the 8440 disks the following comparative analysis was performed and an estimated figure of 45 msec arrived at.

Table 26 compares some of the operating parameters of the Fastrand drum and 8440 disk.

	Fastrand	8440 Disk
Positioning Time	57 m.sec	35 m.sec
Rotational latency	35 m.sec	12.5 m.sec
Word transfer rate	26,283 per sec	138,888 per sec
Average service time	70 m.sec	?

Table 26. Fastrand and 8440 Disk Operating Parameters

The disk is assumed to operate in a manner similar to Fastrand, in that there is a lookahead feature which selects from a queue the request which will cause minimum head movement. However, once the head is in position there is a delay known as rotational latency which is experienced while waiting for the desired record to come under the read or write head. The average value of this delay is given by the time required for the device to rotate one-half of one revolution.

The average service time T_S is given by the sum of the positioning time T_P, the latency time T_L and the data transfer time T_D:

i.e. $T_S = T_P + T_L + T_D$

hence, $T_P = T_S - T_L - T_D$

For the Fastrand drum

$$T_P = 70 - 35 - 10$$

$$= 25 \text{ m.sec}$$

assuming an average record size of 262 words which would require a data transfer time of 10 m.sec. This average positioning time of 25 m.sec is less than the figure of 57 m.sec quoted in Table 25 and is due to lookahead being applied to the Fastrand request queue which results in minimum head movement.

The 45 m.sec service time derived for the 8440 disk consists of a 262 word transfer time of 1.9 m.sec, a latency of 12.5 m.sec, and a positioning time of 30.6 m.sec. This positioning time is only marginally faster than the figure of 35 m.sec given in Table 25; however, it is thought to be reasonable because the disks, being faster than the Fastrand drums, will have smaller queue lengths associated with them which will lead to reduced effectiveness of the lookahead algorithm.

Experiment #10: Three Fastrand Controllers

Experiment #11: Two 8440 Disk Controllers

Experiment #12: Three 8440 Disk Controllers

Ref: Tables 11, 12, 13 and Figure 14

These experiments resulted in reduced processing times of 18.8%, 20.1% and 26.8% respectively. All are typified by increased CPU utilization due to the reduction in I/O channel queue times. The greatest improvement in throughput is obtained when three 8440 disk controllers are employed. It is of interest to note from Figure 14 that in this case all queues are approximately equal, i.e. the queueing delays are equally distributed throughout the system. Such a balanced system, having no significant bottlenecks, uses its resources most effectively; hence its throughput is near optimum for the given set of operating conditions, e.g. a less than optimal core allocation algorithm.

THROUGHPUT MAXIMIZATION EXPERIMENTS

Experiment #13: Core Compaction and Three Fastrand Controllers

Experiment #14: Core Compaction, Three Fastrand Controllers and MAXOPEN = 20

Experiment #15: Core Compaction and Two 8440 Disk Controllers

Experiment #16: Core Compaction and Three 8440 Disk Controllers

Ref: Tables 14, 15, 16, 17 and Figure 15

This set of experiments applied those changes to the system which previous experiments had indicated were most effective in reducing the overall processing time for the given jobmix. The most effective changes were related to an additional Fastrand controller, replacement of Fastrand by an 8440 disk system, core compaction, and an increase in MAXOPEN.

These experiments resulted in reduced processing times of 24.3%, 26.4%, 31.0% and 35.6% due to the increase in core and CPU utilization over the previous set of tests. Figure 15 shows that for the first three experiments of the above set the I/O channel queueing time has again increased and has disturbed the balance achieved in Experiment #12. However, this balance was re-established within the simulator by the addition of another disk controller.

CPU TIME QUANTUM SENSITIVITY EXPERIMENTS

This set of experiments was designed to study the effect of varying the CPU scheduling in an environment of a reduced I/O channel bottleneck and an optimum core allocation algorithm. In addition to the original system baseline configuration, the results of Experiment #13: Core Compaction and Three Fastrand Controllers, is included in Figure 16 for ease of comparison.

Experiment #17: Core Compaction, Three Fastrand Controllers and Double Time Quantum (Ref: Table 18 and Figure 16)

In this experiment, the basic CPU time quantum of 8 m.sec was doubled.

No significant effect was noted.

Experiment #18: Core Compaction, Three Fastrand Controllers and Half-Time Quantum (Ref: Table 19 and Figure 16)

In this experiment, the basic CPU time quantum of 8 m.sec was halved.

No significant effect was noted.

Experiment #19: Core Compaction, Three Fastrand Controllers and Time Quantum Terminating on an I/O Operation (Ref: Table 20 and Figure 16)

The effect of allowing a task to hold a CPU until it requested an I/O operation was studied in this experiment. Run priority was used in place of the switch list level priority of the previous two tests.

No significant effect was noted.

Experiment #20: Core Compaction, Three Fastrand Controllers and CPU Priority as a Function of I/O Frequency (Ref: Table 21 and Figure 16)

In this experiment high CPU priority was given to those tasks having a large amount of I/O activity.

No significant effect was noted.

CONCLUSION ON CPU TIME QUANTUM SENSITIVITY TESTS

It is to be concluded from this set of experiments that for the system configuration and job mix considered, the CPU queue is the least significant (see Figure 16) and hence no advantage can be gained from using the CPUs more effi[illegible]ly.

THIRD SHIFT EXPERIMENTS

A selected set of experiments was performed on the third shift job mix. Their results are discussed below and compared with the corresponding first shift tests.

Experiment #21: Baseline Configuration - Third Shift (Ref: Table 22 and Figure 17)

The total queueing time was 40,467 sec., and the total CPU time 11,525 sec. These figures compare to 43,321 sec and 9,582 sec. respectively for the first shift baseline.

Experiment #22: Zero Tape Mount Time (Ref: Table 23 and Figure 17)

Although the tape mount queue time was eliminated no reduction in overall processing time was observed, due largely to the increase in core queue time.

Experiment #23: Core Compaction (Ref: Table 24 and Figure 17)

In this case the overall effect is an increase in throughput of 3.6% due mainly to CPU and core utilization figures of 53.00% and 77.64% respectively compared to the respective baseline figures of 50.95% and 70.48%. As was the case when core compaction was applied to the first shift mix, the increased user activity causes more rapid requests for I/O

thereby creating an increase in I/O channel queueing times.

Experiment #24: Three Fastrand Controllers (Ref: Table 25 and Figure 17)

This experiment resulted in a significant decrease in I/O queueing times, and an increase in CPU utilization as was observed with the first shift job mix. In this case the throughput was increased by 12.9% compared with 18.8% for the first shift job mix. This difference may well be attributable to the fact that the third shift jobmix consisted of a smaller number of longer running jobs compared to the first shift mix, and hence is more likely to exhibit a larger statistical variation than the first shift mix.

SECTION VI. CONCLUSIONS

It has proved possible to implement a deterministic throughput simulator of the MSFC U-1108/8 system whose System Throughput Characteristic closely matches that of the actual system. The simulation to real-time ratio was approximately 4:1 for the tests carried out.

Based on an initial verification, it is concluded that this model is of sufficient accuracy and efficiency to be of use in determining the effect, on throughput, of varying (a) the 1108 hardware configuration and (b) the EXEC 8 scheduling algorithms.

A set of experiments has been performed in which both hardware and executive changes were simulated. The results from these experiments are the basis for the following recommendations for changes in the present MSFC UNIVAC 1108 system:

1. Add another dual channel controller and reconfigure the FASTRAND subsystems into three banks of four units instead of the current configuration of two banks of six units,

2. Introduce an executive algorithm for compacting core,

3. Increase the value of the MAXOPEN parameter.

Other combinations of changes showed slightly greater improvement in system throughput but these three are the most cost-effective due to the minimal hardware changes. Simulator results show that these changes would increase system throughput by more than 20%, with most of the increase coming from recommendation 1. Recommendations 2. and 3. alone would not greatly enhance system performance and would be effective only if the FASTRAND bottleneck is first removed by recommendation 1.

It is further concluded that the techniques used in the implementation of this simulator may also be applied to other systems, and such models may be used as design tools in determining the optimum hardware/operating system configurations for future computing systems.